2021 年度　卒　業　論　文

Deep Deterministic Policy Gradient を用いたレースゲーム AI 制作効率の向上に関する研究

指導教員：渡辺 大地 教授

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0118303
王　世沢

2022 年 2 月

2021 年度　　卒　業　論　文　概　要

論文題目

Deep Deterministic Policy Gradient を用いた レースゲーム AI 制作効率の向上に関する研究

メディア学部 **学籍番号：M0118303**	**氏名**	王　世沢	**指導教員**	渡辺 大地 教授

キーワード	ルート設計、DDPG、報酬関数、経験再生、 AI

ゲーム産業の発展やゲームの複雑性の向上に伴い、AI 技術を用いたレースゲームの訓練問題は重要になってきている。本論文では、レーシングカーのトレーニングの質とスピードを向上させるために、DDPG を最適化する。具体的な研究内容は以下のとおりである．1）動的報酬関数に基づく DDPG ルート設計アルゴリズム：DDPG ネットワークに畳み込みニューラルネットワーク部分を追加するため、訓練時期によって、ネットワークに対する需要が異なる。トレーニングの前段階では、レースが真の経路に沿った形をとることができないため、レースの中心線からの位置と衝突回数の 2 つのパラメータに大きな係数を設定することで、高い報酬を得ることができる。常時ネットトレーニングの成熟化で、後期レーシングカーはドリフトや衝突など難易度の高い操作を行う必要がある。このとき係数の重みを下げる必要がある。2）異なる重みの経験再生に基づく DDPG ルート設計アルゴリズム：DDPG モデルの中で、経験再生はモデルの重要な部分であり、モデルの経験は同じ重み値を持っているため、訓練過程の中で経験の重要度を際立たせることができないため、本文は異なる経験に異なる重みを与えて、ネットに対して訓練を行う。本論文では、モデルを 100 回、500 回、1000 回、2000 回はトレーニングしたが、トレーニング回数が 2000 回であれば、モデルの報酬値は約 60000 前後のレベルで安定した。

目 次

図 目 次

第 1 章

はじめに

1.1 研究の目的と意義

近年、深層強化学習を用いて、多くのゲーム AI が自動的にゲームルールを学習するようになって来ている 。2015 年に Google 社が『Nature』誌に Deep Q-network の改良版を発表して以来、深層強化学習はさまざまなゲームに応用されている。多くのゲームでプロプレイヤーと同等以上の性能を有するプレイヤ AI の制作に成功している [1]。深層強化学習は Atari 2600 の 49 種類のゲーム中 43 種類で従来の人工知能による得点を上回り、29 のゲームではプロプレイヤーと同等またはそれ以上のパフォーマンスを見せた。特にブロック崩しでは、400 回プレイするとボールの取りこぼしがなくなり、600 回のプレイの後には次々と攻略法を生み出し、高得点を取るようになった。またもっとも上達したピンボールでは人間の 25 倍のスコアを取った [2]。

レースゲームの開発は近年複雑になって来ている。レースゲームの作成において、強化学習を用いた AI は作成作業効率を上げる手段としても利用されている。しかし、試作コースに対しての強化学習トレーニング速度は遅く、最終的にコースに適した高性能な AI が作成できるまでには十数日から数ヶ月かかる。レースゲームの種類によっては、その都度 AI のパラメータ調整が必要であったり、レースゲームを作成した後でないとコースとしての面白さが判断じづらい等の理由から、レーシングカー AI がコースへ迅速に適応する事が望ましい。レーシングカー AI がコースに適応していく過程からは、コースの難易度調整への応用も期待できる。これらの複合的理由から、レーシングカー AI をゲーム制作効率向上に利用することが可能であると言える。本研究では、レーシングゲームにおけるレースコースデザインの効率化に着目した。

ゲーム中のルート設計では、レーシングカーの進行方向に障害物や選択可能なルートが現れる可能性があるが、深層決定方策勾配（Deep Deterministic Policy Gradient、DDPG）[3] によってレーシングカーの連続働作 [4] の情報を提供できた。DDPG がレーシングカーに提供する障害物回避アドバイスと経路アドバイスに基づいて、本研究では DDPG を改良しレーシングカーの走行効率を大幅に向上させ、レーシングカーの安全走行を効果的に補助することを目的とする。提案手法により、レーシングカーはより速くゲームコースの状況に適応することができ、同時にコースの最適経路や理想的なレコードタイムといったコース自体の出来栄えをテストする事ができ、ゲーム全体の制作効率を高めることができる。従来の DDPG アルゴリズムを用いた学習と比較し、本提案手法である改良 DDPG アルゴリズムは、少ない学習回数でより良い走行結果を得ることができた。

1.2 先行研究

ルート設計や障害物回避の問題の研究は多数ある。Arumugam ら [5] は、移動する物体の経路を記述する大量のデータへの応用において、物体間の接続性の問題を考慮して、Closest-Point-Of-Approach の特定の接続方式を研究し、基礎データの特性に応じて計算接続方式を変更する新しい適応的接続アルゴリズム（Distance of Closeset Point of Approach，DCPA）を提案した。Milios ら [6] は、n 個の頂点のグラフに対して、すべての頂点対を 1 つの辺で接続し、辺ごとに一定の正の長さを持つ Dijkstra 探索アルゴリズムを提案した。アルゴリズムは、n 個の頂点間の全長最小問題と、与えられた 2 つの頂点間の全長最小問題を主に解決する。Hart ら [7] は、ノード n から目的ノードへの真の最小コストの下限、すなわち目的ノードへの経路があれば、最小コストの経路を見つけることができ、ヒューリスティック関数に対しても一貫性の仮定を満たすことができることを証明している。以上の経路探索アルゴリズムは、グラフ探索に基づくルート設計アルゴリズムである。

人工知能技術の発展に伴い、Smierzchalski ら [8] は遺伝的アルゴリズムによって経路を計画し

衝突の問題を解決した。提案された手法は、特定の静的および動的環境における最適な安全経路を計算することを可能にし、機動領域の特定の境界、ナビゲーション障害物および他の移動目標の情報を考慮することにより、衝突回避問題を静的および動的制約を伴う最適化問題に単純化する。この方法により、船舶の安全運行軌跡を算出することができる。Lee ら [9] はポテンシャル場法を改良したファジィ論理衝突アルゴリズムに基づいて、静的障害物と動的障害物の識別を実現した。この改良された方法は、トラックの保持と障害物への衝突回避とを達成することができる。レールを保持することにより、走行状態を安定させることができ、手間を省くことができる。衝突回避機能は、静的物体と動的物体を識別することにより、走行経路を計画することにより障害物を回避する。Zhuo ら [10] は、自己学習ニューラルファジィネットワークを提案することにより、新しい知識を適応的に学習し、古い知識を忘れることなく、衝突能力の精度を向上させるアルゴリズムを提案した。不正確な情報を扱うために、モデルにファジーセット解釈措置を追加した。また、ニューラルネットワーク構造を用いてファジィ推論システムのパラメータを訓練し、ハイブリッド学習アルゴリズムとオフライン訓練データに基づいてネットワークを訓練した。このモデルは、正確な衝突回避判定を行うことができ、データ処理の負担を軽減することができる。Jha ら [11] はファジィ推論とエキスパートシステムを統合して衝突防止システムに応用した。この方法のハイライトはニューラルネットワークを利用して衝突のリスクを計算し、KT 方程式を利用して DCPA と TCPA 関数を確定し、適応ネットワークファジィ推論を用いて推論表を再設計したことにある。最後に多層感知器ニューラルネットワークを衝突防止システムに応用し、ファジィ論理を補う。Szlapczynski ら [12] は、ゲーム理論の仮説と進化規則を応用した衝突回避手法を提案した。制約の最適化問題は、遭遇したすべての状況の軌跡セットを探索することとして定義される進化的アルゴリズムによって解決される。最適化アルゴリズムを設計する過程で、特に衝突回避演算子に対して、得られた軌跡が最も安全で、平均損失が最小になるように設計した。Watkins ら [13] は、計算に対する要求が限定された動的プログラミングのインクリメンタル手法である Q-learning アルゴリズムを提案しており、特定の状態における特定の行動の質の評価を連

続的に改善することで効果を発揮している。著者は Q-learning の収束定理を詳細に示し、すべての行動がすべての状態でサンプリングを繰り返し、行動値が離散的に表される限り、Q-learning は 1 の確率で最適行動値に収束することを示した。Stutzle ら [14] は、アリの餌探し（ランダム探索）とフェロモン通信（情報交換）の考えに基づいて有力な経路を形成する組合せ最適化のためのグループメタヒューリスティックに基づく探索アルゴリズムを提案し、グラフ中で最適な経路を探索する確率型アルゴリズムであり、多くの点で広く利用されている。Lei ら [15] は改良された 3 次元アリ群アルゴリズムを提案し、アルゴリズムの実行効率をさらに高めるために、アリ群アルゴリズムと遺伝的アルゴリズムを結合し、新しい経路最適化アルゴリズム（AC-GA）を提案した。この方法は、情報伝送路の最適化に適用され、最適化プロセスにおける不確実性と複雑性を解決する。深層決定方策勾配（Deep Deterministic Policy Gradient、DDPG）は、勾配アルゴリズム強化学習の決定論的策略を考慮する。決定論的策略勾配は、行動－価値関数の予想勾配であり、通常のランダム策略勾配よりも効率的である。このアルゴリズムは、十分な探索を確保するために、探索的行動策略から特定の目標策略を学習する非策略的行為者批判アルゴリズムを導入する。同時に、このアルゴリズムは高次元空間においてそのランダム対応のアルゴリズムより明らかに優れていることを証明した。強化学習と経路計画におけ研究結果に基づいて、本論文はレーシングカーのトレーニングの質とスピードを向上させるために、DDPG を最適化する。本文で提案したアルゴリズムがトレーニングの速度の方面比較的に良い効果を持つことを証明した。

1.3 論文の構成

本論文の構成は、2 章ではパスプランニングに関する理論である、強化学習、DQN、DDPG について述べる。3 章では、動的報酬関数に基づく DDPG ルート設計アルゴリズムと、異なる重みの経験再生に基づく DDPG ルート設計アルゴリズムについて述べる。4 章では実験結果の分析について述べる。

第 2 章

ゲームレーシングパス理論の基礎

本章では、強化学習の概要及び、本研究で利用する DDPG について述べる。

2.1　強化学習の概要

強化学習（Reinforcement Learning、RL）[16] は機械学習の一種であり、実際の問題において環境とのインタラクションによって報酬を最大化する問題を解決するために用いる。強化学習は訓練のためのデータを必要としないため、このアルゴリズムは問題に対する推論能力を持ち、ある程度のロバスト性を持つ。図 2.1 は強化学習の模式図である。

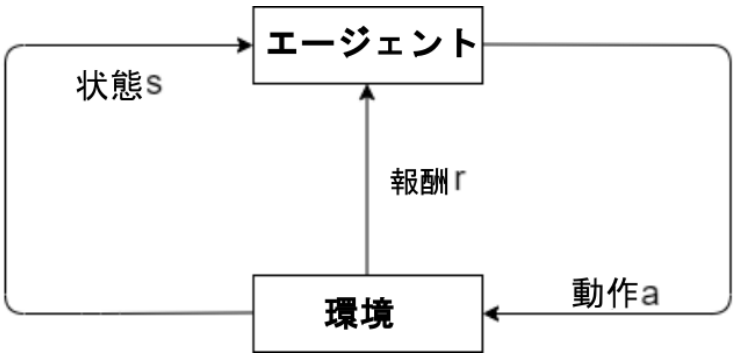

図 2.1　強化学習の模式図

2.1.1　マルコフ決定過程

マルコフ決定過程（Markov Decision Process、MDP）[17] はマルコフ性を持つ数学モデルである。最初に提案された離散時間マルコフ決定過程は離散ランダム最適制御モデルに基づいて得られたものであり、後の学者はその動的計画方法を改善し、推薦システムや RL などの分野に広

く応用されている。マルコフ性では、現在時刻の状態は、前時刻の状態および動作のみに関連し、残りのすべての時刻の状態および動作とは互いに独立している、すなわち、現在時刻の状態および動作は、前時刻の状態および動作のみに影響されると規定されている。この状態はマルコフ性を持つと考えられ、式 (2.1) は状態遷移式を表す。

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1 \cdots S_T] \tag{2.1}$$

マルコフ決定過程は制御マルコフ連鎖（controlled Markov chain）とも呼ばれる、ある state において、1 つの action を発生し、1 つの reward を発生し、次の時刻の state を状態遷移確率関数によって決定するように記述することができる。環境はイベントに対して状態遷移関数と報酬関数を提供することができ、これによりマルコフ決定過程を構成することができる。この過程では、時間とともに状態が遷移する、すなわち状態間の相互遷移が発生する。マルコフ決定過程は決定付きマルコフ報酬過程であり、この環境ではすべての状態がマルコフ性を持つ。マルコフ決定過程とマルコフ報酬過程の違いは、マルコフ決定過程に集合 A が付加されている点である。ここで、5 タプル $< S$、A、P、R、$\gamma >$ を用いてマルコフ決定過程を表す。

- S は有限状態の集合を表す。
- A は動作の有限の集合を表す。
- P は状態遷移確率行列を表し、$P^a_{ss'} = P[S_{t+1} = s' \mid S_t = s,\ A_t = a]$。
- R は報酬であり、$R^a_s = E[R_{t+1}|S_t = s, A_t = a]$ と定義され、現在時刻 t の状態 S_t 表わす、仕事 $A_t = a$ をとった後、次の時刻 $t + 1$ に獲得できる期待報酬を表す。
- γ は割引因子（discount factor)、$\gamma \in [0, 1]$ である。

図 2.2 は、$(s_0, a_0, r_1, s_1, a_1, r_2, \cdots, s_n, a_n, r_{n+1})$ で表現できるマルコフ決定過程を示している。

マルコフの意思決定過程では、エージェントと環境との間で対話的なプロセスを行う。この過程においてエージェントは学習者や意思決定者として、環境はエージェントとインタラクションを行う他のすべての物体として機能する。時刻 t において、エージェントの境状態 s_t は、エー

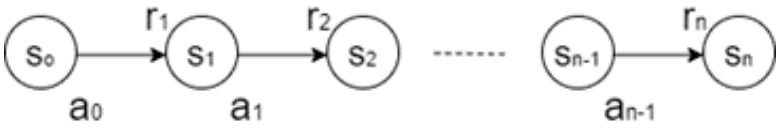

図 2.2　マルコフのランダム過程

ジェントが意思決定行動を行うことによって、それに応じた動作 a_t を行うことである。このとき環境は遷移確率 P によって次の状態 $+1$ を得ることができ、環境から報酬 r_{t+1} を得ることができる。

2.2　DQN アルゴリズム

DQN アルゴリズム [18] は Q-learning を基に、ニューラルネットワークを用いて Q 値をフィッティングすることで、Q-table 次元の制約により高緯度状態に適用できない問題を解決した。ニューラルネットワークのトレーニングプロセスにおいて、目的は、損失関数が最小値、すなわち予測ラベルと真のラベルとの間の差を得ることである。このプロセスは、トレーニングのために大量のデータを必要とし、その後、逆伝播を使用してネットワークのパラメータを更新する。したがって、Q ネットワークを訓練する過程では、一定数のラベル付きサンプルが必要となる。このうち Q ネットワークにラベル付きサンプルを提供する方法が Q-learn アルゴリズムである。図 2.3 は DQN ネットワークである。

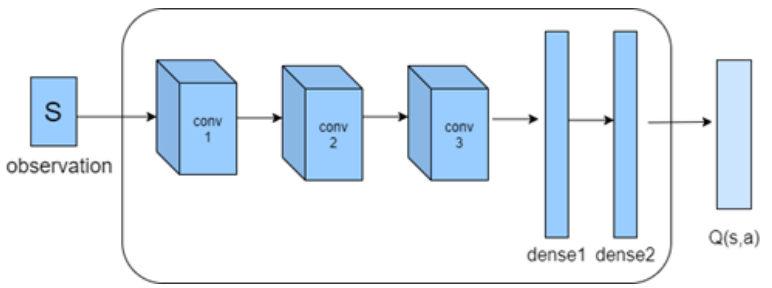

図 2.3　DQN ネットワーク

環境からネットワークへの引数として observation を与え、エージェントは値関数ネットワークから $Q(s,a)$ を得る。エージェントの行動結果に応じて、エージェントは reward を受け取り、

行動結果が環境の状態を変化させる。変化した状態から次の observation を受け取る。以上繰り返す事で、より効果の高い値関数ネットワークを作成していく。値関数ネットワークの最適化を図る際、予測値と正解値の差を少なくする。この差を求め関数を損失関数と呼び、損失関数による出力値が最小になることを目指す。損失関数は式 (2.2) を用いる。

$$L_i\left(\theta_i\right) = E_{(s,a,r,s^i)\sim U(D)}\left[\left(r + \gamma_{a'}^{\max} Q\left(s', a^i; \theta_i^-\right) - Q\left(s, a; \theta_i\right)\right)^2\right] \tag{2.2}$$

式 (2.2) では、θ_i^- は i 回トレーニング後の target ネットワークパラメータ、θ_i は Q ネットワークのパラメータ、を示す。更に最急降下法を用いてより損失が少なくなるパラメータを逐次的に求めていく。式 (2.3) は式 (2.2) の θ 勾配を求める。

$$\frac{\partial L_i\left(\theta_i\right)}{\partial \theta_i} = E_{(s,a,r,s^i)\sim U(D)}\left[\left(+\gamma_{a'}^{\max}\left(s', a^i; \theta_i^-\right) - Q\left(s, a; \theta_i\right)\right) \nabla_{\theta_i} Q\left(s, a; \theta_i\right)\right] \tag{2.3}$$

トレーニング中、毎回トレーニングの $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ を記憶プールに入れる。

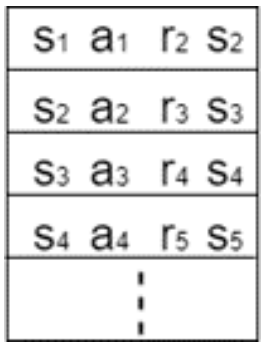

図 2.4　replay memory

経験再生 (experience replay) の考え方を DQN ネットワークに導入する。このアイデアはある時点における環境の状態を複数保存しておき、時系列をランダムにサンプリングしてニューラルネットワークの学習データに用いるというものである。経験再生の使用には主に 2 つの理由がある：1）ニューラルネットワークは教師あり学習のモデルで、トレーニングのデータは独立分布であることが求められる；2）Q-Learning プログラムの場合、時系列順なデータをインプットデータにすると、時系列の特徴が強く反映されてしまうため、経験再生によってデータの関連性を排除する。これをゲーム作成の場合で考えた場合、インプットデータはゲームの連続する描画フレー

ムから抽出されるため、インプットデータは非常に時系列による関連性が高くなる。経験再生を適用しなければ最急降下法による更新時にある一定方向にしか収束しないため、損失関数の最適化には不最適化となる。

DQN ネットワークでは、今の時間 t、状態 s の条件で、行動 a を選択するとき報酬は r_t とし、式 (2.4) で求める。

$$r_t = \begin{cases} 1 & \text{increase} \\ 0 & \text{noexchange} \\ -1 & \text{decrease} \end{cases} \tag{2.4}$$

長期蓄積割引報酬は式 (2.5) で求める。

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{2.5}$$

2.3 DDPG アルゴリズム

DDPG は 2016 年にグーグルのチームによって提案したアルゴリズムである、このアルゴリズムは Actor-Critic アルゴリズム [19] の枠組みを使い、DQN アルゴリズムの思想を参考に連続動作空間問題を解決している。

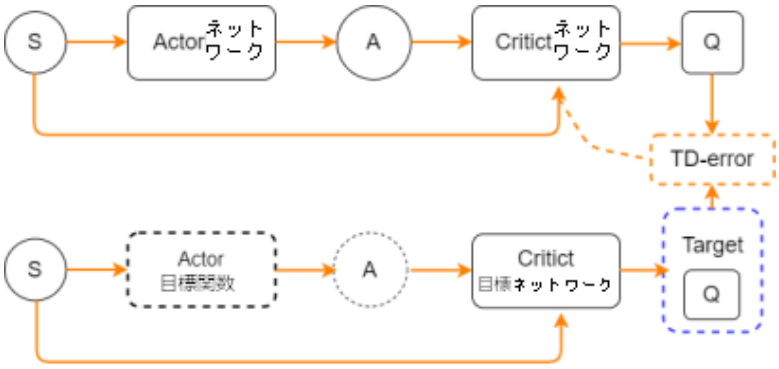

図 2.5　DDPG 工程図

図 2.5 は DDPG 工程を示す。Critic ネットワーク使って Q を推測する。このネットワーク入力は二つ：行動と状態、行動と状態は Critic ネットワークに入力する必要がある。このネットワークは TD-error を損失値とする。Actor ネットワークは、ネットワークの出力は行動とする。

ネットワークの機能は一つの動作 A を出力する。A が Critic ネットワークに入力された最大の Q を得る。Actor の更新方式は最急上昇法の原理を用いてネットワークを更新を行う。Actor-Crite 枠組みの更新方法は一方行動評価関数モデルを学び、サンプルから算出された行動価値に迫る、一方で価値の高い動作の出力可能性を高める方策関数モデルを学習する。

DDPG アルゴリズムは DPG（Deterministic Policy Gradient）アルゴリズムの基礎の上で発展してきたので、DDPG アルゴリズムをよりよく記述するために、ここで DPG アルゴリズム [20] 理論を簡単に記述する。DPG は決定性のある行動方策であり、各ステップの値は関数 μ によって決定された値を得ることができる。

$$a_t = \mu\left(s_t|\theta^{\mu}\right) \tag{2.6}$$

確実性のある策略が必要なのは、PG 法には二つ欠点がある。（1）PG を用いてランダム方策を獲得した後でも、1 ステップ動作ごとに、得られた最適方策分布をサンプリングして動作の具体値を獲得する必要があり、動作値は一般に高次元のベクトルであり、多くの計算力を消費するという欠点がある。（2）PG の学習過程では、ステップごとの方策勾配の計算過程において、動作空間全体にわたって積分を行う必要がある。

DDPG アルゴリズムはディープニューラルネットワークを DPG アルゴリズムに融合したものであり、主な改善点は畳み込みニューラルネットワークを用いて方策関数 μ と Q 関数をシミュレーションし、それからディープラーニング方法を用いてニューラルネットワークを訓練することである。Q 関数の実装およびトレーニング方法は、DQN 方法を採用している。

DDPG アルゴリズムの中で、方策ネットワークは 1 つの畳み込みニューラルネットワーク [21] を用いて μ 関数に対してシミュレーションを行い、この方策ネットワークのパラメータは以下である；強化学習の訓練過程では、行動方策 β にランダムノイズを導入して、潜在的により優れた方策を探索する。ここで注意すべきことは、行動方策 β は求められる最適方策ではなく、この関数の役割は、状態遷移やイベントの歩行経路などを含む必要なデータセットを得るために訓練過程で環境の動作を生成することである。次にこれらのデータセットを用いて方策 μ を訓練し、最

適な方策を獲得する。

Q 関数は action-value 関数であり、状態 s_t で動作 a_t を行った後、方策 μ を継続的に実行した場合に得られる期待値は Bellman の等式で定義できる。

$$Q^{\mu}(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_t + 1))] \tag{2.7}$$

式 (2.7) から、Q 関数は再帰式であることが分かるが、実際にはステップごとに Q 値を計算することは現実的ではないため、Bellman の式を関数でシミュレーションすることで解決している。DDPG アルゴリズムでは、畳み込みニューラルネットワークを用いて Q 関数をシミュレーションするため、このネットワークを Q ネットワークと呼び、パラメータを θ^{Q} とする。

方策 μ の性能を測定する際には、関数 J を用いて測定し、この行為を performance objective と呼び、off-policy シーンに対して評価関数を式 (2.8) に定義する。

$$J_{\beta(\mu)} = \int \rho^{\beta}(s) Q^{\mu}(s, \mu(s))\, ds = E_{s\sim\rho^{\beta}}[Q^{\mu}(s, \mu(s))] \tag{2.8}$$

式 (2.8) において、s は環境状態を表し、これらの状態はイベントの行動方策に基づいて生成され、分布関数は ρ^{β}、$Q^{\mu}(s, \mu(s))$ は状態ごとに方策 μ に従って動作を選択できた場合に生じる Q 値、すなわち $J_{\beta(\mu)}$ は s が ρ^{β} 分布に従っている場合、$Q^{\mu}(s, \mu(s))$ の期待値である。

アルゴリズムの訓練目標は、$J_{\beta(\mu)}$ の値を最大化し、Q ネットワークにおける損失値を最小にすることである。最適行動方策 μ の定義は $J_{\beta(\mu)}$ を最大化する方策である：

$$\mu = \underset{\mu}{\operatorname{argmax}}\, J(\mu) \tag{2.9}$$

方策 μ を求める過程は、μ ネットワークパラメータの最適解を求める過程である。

2.4 本章のまとめ

本章では、マルコフの決定過程、方策に基づく強化学習、値関数に基づく強化学習を紹介した。強化学習の基本的な概要を含め、ルート設計の理論的基礎を中心に紹介した。続いて、DQN モデ

ルと DDPG モデルを紹介した。2 つのモデルは思想上で受け入れ性があり、つまり DDPG は部分 DQN のアルゴリズム思想を参考する、それによって連続動作空間問題を解決した。

第 3 章

DDPG アルゴリズムの改善に基づくゲームレーシングパス計画の研究

DDPG アルゴリズムは確定的方策方法を通じてレーシングカーなどの運動物体に合理的な道路計画と障害物回避の提案を提供することができる。しかし、実際の運用において DDPG アルゴリズムにも一定の限界が存在する。本論文は DDPG アルゴリズムを改善することによって、アルゴリズムの効果がより実際の需要を満たすようにした。本論文は主に 2 つの方針の組み合わせで DDPG に対して改善を行う：まず報酬関数 [22] に基づいて DDPG に対して改善を行い、動的な形式を採用してそれに対して改善を行う。次に、経験再生重み付けの形で DDPG を改善する。

3.1 動的報酬関数に基づく DDPG アルゴリズムの改善

強化学習はマルコフ過程で定義される連続意思決定問題であり、多くのタスクの状態 – 動作空間において報酬信号が 0 となる現象を報酬関数のスパースネス（sparsity of reward）と呼ぶ。疎な報酬関数ではアルゴリズムの収束が遅く、イベントは環境と何度も作用し、大量のサンプルを学習して最適解に収束する必要がある。

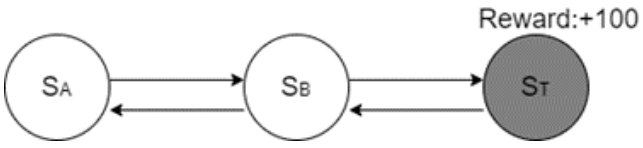

図 3.1　状態遷移報酬

初期訓練過程では、イベントは S_A 状態から S_T 状態への過程で 2 段階（または 2 つ）を経る必要があるので、S_T に到達できる確率は 1/4 である。図 3.1 は、状態が少ない場合、あるいは状態

が単純な場合の模式図を示しているが、状態-行為空間が大きい場合にはイベントが S_T に到達できる確率が大きく低下し、報酬が 0 になる結果となっている（非終点状態報酬はすべて 0 であるため)。式 (3.1) は、目標に最初に到達する確率をイメージしたものである。

$$P = \frac{1}{|A|^M} \tag{3.1}$$

ここで、M はゴールまでの歩数、A はエージェントがとりうる行為の数を表す。

3.1.1 報酬関数

エージェントがゴールを探す過程では、状態-行為の影響に制限されるため、ゴールに到達する確率は小さい。このような状況を踏まえて、図 3.2 に示すように、エージェントを探す過程で追加的な報酬を与えることができる。

図 3.2　ボーナス

図 3.2 では、エージェントが S_A と S_T の間で遷移を続けている場合、遷移ごとに 1 ポイントの報酬が獲得されており、これは明らかに付加報酬を設計する本来の目的に反している。そのため、ポテンシャルエネルギー関数を導入してこのような問題を解決した。

ポテンシャル関数は物理学の概念を参考にしており、エージェントが 1 つの高ポテンシャルエネルギーから低ポテンシャルエネルギーに移行するとき、追加の報酬（reward）を得る。逆に、エージェントが低ポテンシャルエネルギーから高ポテンシャルエネルギーに移行した場合、報酬は消失する。2 つの状態のポテンシャルエネルギーの差を追加の報酬として使用することで、MDP の最適解が変更されないことが保証する。

$$R'(s, a, s') = R(s, a, s') + F(s') \tag{3.2}$$

ここで、式 (3.2) を示す $R'(s, a, s')$ は、変更後の報酬関数である。

$$F(s') = \Phi(s') - \gamma\Phi(s) \tag{3.3}$$

ここで、式 (3.3) を示す γ は値引因子、F はポテンシャルエネルギー関数である。

3.1.2 動的報酬関数

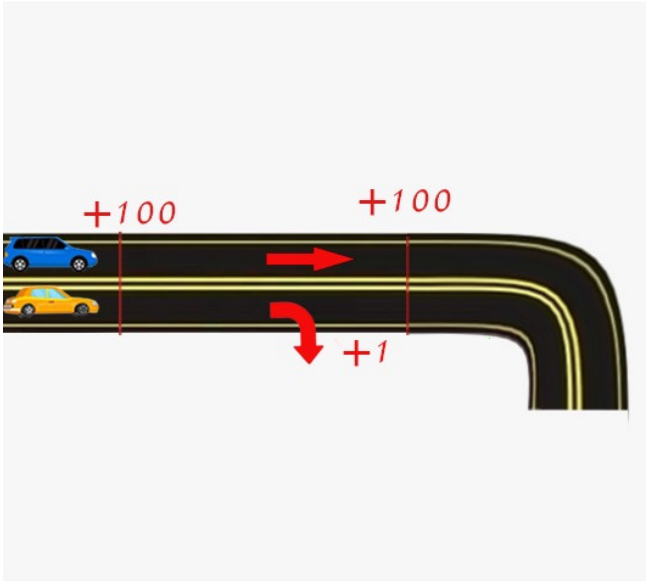

図 3.3 動的報酬関数の判断方法

DDPG アルゴリズムに畳み込みニューラルネットワークを追加して大量のサンプルを訓練するので、訓練過程は何度も繰り返す必要がある。訓練過程の初期では、大量の情報やロジックを学習しきれず、レーシングカーがサーキットに沿って走行出来ない、あるいは逆走する等してしまう。したがって、初期段階の報酬関数では、距離中心線位置と衝突回数の二つの係数を大きく設定することで、モデルがこれら二つのパラメータに注目するようにした。しかし、訓練過程の後期では、カーブを通過するためにはドリフトと衝突が必要で、中心線から離れて衝突する必要がある。したがって、これら 2 つのパラメータの係数を小さい値に設定する必要がある。前後二段階では、異なる時期ごとに対応するため、コース道路中心線からの位置と衝突回数の二つの係数の設定を変更する。本研究の動的報酬関数の判断方法は図 3.3 ように、青いレーシングカーが 2

つチェックポイントを通ることで報酬 +200 獲得できるのに対し、黄色レーシングカーが 1 つ目のチェックポイントを通過し報酬 +100 獲得し、方向変更に応じて報酬 +1 獲得できる。追加的な報酬を与えることで、より速くレーシングコースの探索が行える。

3.2 経験再生重みに基づく DDPG アルゴリズム

RL アルゴリズムがデータを学習している間、エージェントは観測された経験に基づいてパラメータを漸増的に更新することができる。通常、エージェントはパラメータを更新した直後にデータを破棄する。しかし、これには 2 つの問題がある。まず、強い相関関係を持つデータがランダムグラディエントアルゴリズムで必要とされる仮定を破っていることである。次に、廃棄されたデータは有用な可能性もある。

経験再生は、経験を再生メモリに格納し、最近の経験を混合することで時間依存性を減らすことができるとともに、希少な経験も繰り返し更新されるという 2 つの問題を解決する。したがって、経験再生は学習に必要な経験の数を効果的に減らすことができる。一般的に言えば、経験プレイバックとは、経験を抽出する際に最も価値のある経験を優先的に抽出することであるが、抽出する経験は多様性を備えていなければならず、そうでないと過適合の問題が生じる。理想的な状態は、価値の高いものほど抽出される確率が高く、価値の低いものも一定の確率で抽出されなければならない。DQN アルゴリズムでは、経験の価値を式 (3.4) で測定する。

$$Q_\omega(s_t, a_t) = Q_\omega(s_t, a_t) + [r_{t+1} + \gamma \max_a Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t)] \tag{3.4}$$

ここで、TD-error は式 (3.5) を示す。

$$\delta_t = r_{t+1} + \gamma \max_a Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t) \tag{3.5}$$

最適化において、目標は TD-error の値をできるだけ小さくすることであるが、もし TD-error が比較的大きい場合、我々の現在の Q 関数が目標の Q 関数からかなり離れていることを意味しており、この場合、値ができるだけ小さくなるように更新を続けるべきである。そのため TD-error では経験の価値を測ることができる。

経験値の高い経験だけをサンプリングしてモデルがオーバーフィッティングされることを避けるために、TD-error の値が 0 でも確率的に抽出されるように、確率的に経験抽出を行う。次に、各経験の優先度を示す。式 (3.6) は各経験抽出確率判断方法である。

$$P(i) = \frac{p_i}{\sum p_i} \qquad (p_i = |\delta_t + \epsilon|) \tag{3.6}$$

ϵ は小さな値で、TD-error が 0 になることを防ぐ。

DDPG の経験再生プールでは、経験再生ごとに重みがなく、状態遷移の前後で経験の重要度が同じである。本論文では、式 (3.4) を用いて各経験の価値を測ることで経験の重み情報を得ることができ、経験の再生ごとに異なる重みを与えることで重要度の異なる経験情報を得ることができる。

3.3 本章のまとめ

本章では、主にゲームのレースの経路における DDPG アルゴリズムの計画を紹介し、DDPG のルート設計問題に基づいて、このアルゴリズムに 2 つの面の改善を行った。まず強化学習中の伝統的な静態奨励関数の弊害に対して、本文はヒューリスティックアルゴリズムの思想に基づいて、静態奨励関数に対して改善を行い、異なる訓練過程に対して異なる奨励関数を採用し、このようにニューラルネット訓練の論理とレースの最適化の需要に符合する。次に DDPG における経験抽出を改良すると、オリジナルの経験は重み情報を持たない、すなわち各経験の重みは同じであり、各経験が生み出す価値が異なるため、これらの経験に異なる重み情報を付与する必要がある。

第 4 章

実験結果と分析

本章では、提案手法の評価実験及び実験結果に対する分析について述べる。

4.1 実験環境の説明

表 4.1 は実験シミュレーション環境を表す。実験シミュレーションプラットフォームは、python 2.7、Tensorflow 0.10、gym 0.16、Opencv3.0、h5py2.7、Keras 1.1 を用いた。

表 4.1 実験シミュレーション環境

OS	Ubuntu 18.04
CPU	AMD Rezen 5 5600X
メモリ	32.0GB
GPU	AMD Radeon™ RX 6700 XT

4.2 アルゴリズムモデルの訓練

本節は提案した動的報酬関数及び異なる重み経験再生に基づく DDPG ルート設計アルゴリズムに対して実験分析を行う。従来の DDPG と提案手法である改良 DDPG を同一の実験条件で比較することにより、提案アルゴリズムの有効性を示す。それぞれの手法で 2000 回トレーニングを行なった結果を示す。

図 4.1 は、トレーニング回数が 100 回（episode = 100）迄の、レースカーの各ラウンドのシミュレーショントレーニングで得られた総リターン値を分析したもので、トレーニング回数を横軸に、反復ごとに得られた総リターン値を縦軸にした折れ線グラフをプロットしたものである。

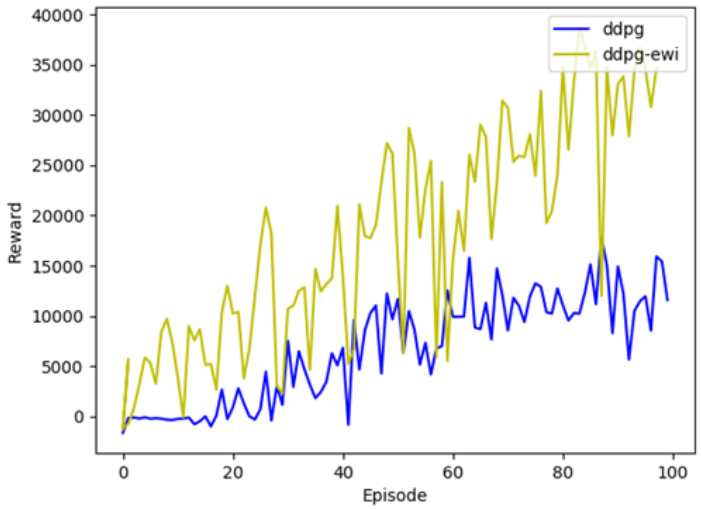

図 4.1　episode = 100

シミュレーション実験を行った結果、反復の過程の中で、2 つのアルゴリズムの総リターン値は徐々に上昇の傾向にあり、しかも上昇の傾向の中に揺らぎ上昇の要素が存在した。改良された DDPG アルゴリズムは、改良前の DDPG アルゴリズムよりも高速に目標リターン値に到達できることが分かった。改善前の DDPG アルゴリズムは、最初の 20 回の反復の中でリターン値がほとんど変化せず、0 の位置にあるのに対し、改善後の DDPG アルゴリズムのリターン値は急速に変化することができるが、振動の成分が大きい。一方、両アルゴリズムを全体的に見ると、改良前の DDPG は反復過程でのリターン値の変化率が小さく、緩やかに上昇する過程にあるのに対し、改良後の DDPG は反復過程でのリターン値の変化率が大きく、ある値に迅速に到達することができる。改良前の DDPG アルゴリズムは 60 ラウンド前後でリターン値がすでに緩やかな状態にあり、この時のリターン値は 15000 前後である、改良後の DDPG アルゴリズムは 85 ラウンド前後でリターン値が緩やかな状態にあり、この時のリターン値は 38000 前後であり、レーシングカーは各ラウンドで比較的短い時間で目標エリアまで走行することができる。改良された DDPG アルゴリズムでは、トレーニング回数が 50 前後で 30000 付近のスコアが得られており、次の反復では、各反復レーシングカーがサーキットの周りを 1 周することができることを示しており、

レーシングカーがゲームのサーキットにフィットし続け、サーキットの道路状況情報により迅速に適応することができることが示されている。改良前の DDPG アルゴリズムと比較して、このアルゴリズムはトレーニングの効率を大幅に向上させる。

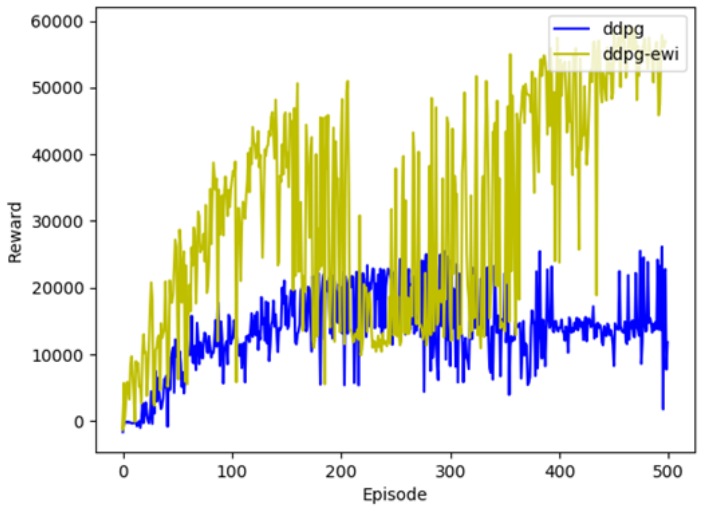

図 4.2　episode = 500

図 4.2 はトレーニング回数 2000 回のうち、0 回から 500 回（episode = 500）迄のデータをプロットしたものである。上昇の傾向の中に振動上昇の要素が存在して、観察によると改良後の DDPG モデルは 200 回のラウンドの時に 1 つの比較的に大きい振動が存在して、リターン値の交差幅は 40000 に達している。改良前の DDPG アルゴリズムは 400 回のラウンドの時に 1 つの比較的に大きい振動が現れて、しかもこの振動は 60 個のラウンドほど持続して、振動幅は約 8000 あった。改良前の DDPG アルゴリズムは、ラウンド数が 150 回に達すると立ち上がりの速度が緩やかになり始める。改善された DDPG アルゴリズムのリターン値は急速に変化する。0 から 150 ラウンドの中で、改善された DDPG はリターン値が 50000 の位置に急速に上昇した後、150 から 200 ラウンドでは、リターン値は不変であった。200 から 250 回のラウンドの中で、リターン値が再度急激に変化し、リターン値は 50000 から 10000 迄低下した。リターン値は 250 ラウンド後に 40000 の位置まで急速に引き上げられ、60000 の位置までゆっくりと成長し始める。改良

前の DDPG アルゴリズムは 150 ラウンド前後で、リターン値はすでに緩やかな状態にあり（ここでは急激に変化の部分は考慮しない)、この時点でのリターン値は 20000 程度であり、改良後の DDPG アルゴリズムは 370 ラウンド前後でリターン値は緩やかな状態に達し、この時点でのリターン値は 50000 程度であり、レーシングカーは各ラウンドで比較的短い時間で目標エリアまで走行することができる。トレーニング回数が 450 回の場合、得点は約 60000 である。次の反復では、トレーニングごとにレーシングカーがコースを 2 周することができ、トレーニングの効率がさらに向上したことを示している。

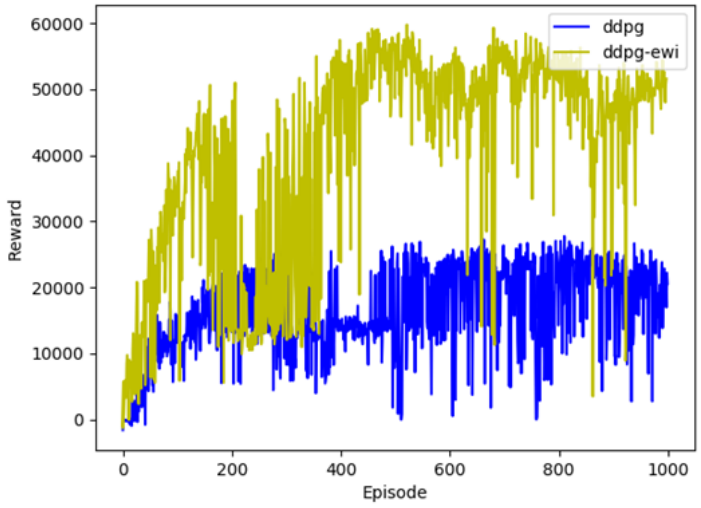

図 4.3　episode = 1000

図 4.3 は、トレーニング回数 2000 回のうち、0 回から 1000 回（episode = 1000）迄のデータをプロットしたものである、トレーニング回数が 1000 の場合、改善前後の DDPG アルゴリズムの効果を全体として反映することができた。全体的な分析から見ると、両アルゴリズムに急激に変化の部分があり、かつ改良前の DDPG アルゴリズムは迅速に安定な状態に達することができたが、リターン値は比較的低いレベルにあった。改良された DDPG アルゴリズムは、各ラウンドにおいて高いリターン値レベルを達成することができ、改良された DDPG アルゴリズムがより短い時間で目標エリアまで走行することができることを示している。

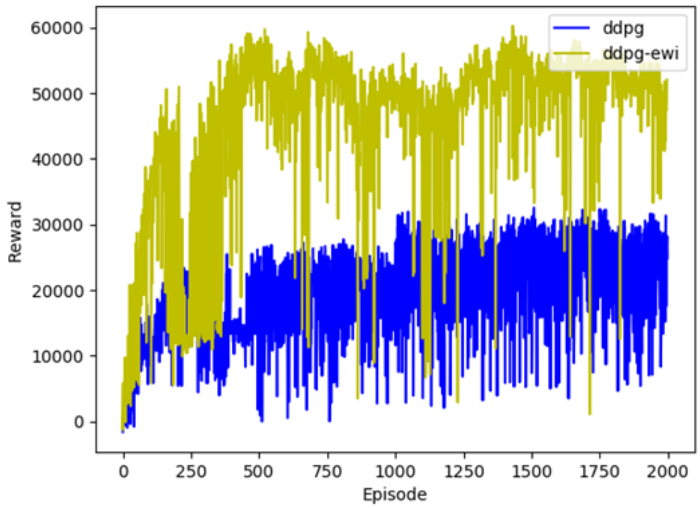

図 4.4　episode= 2000

図 4.4 は、トレーニング回数 2000 回のうち、0 回から 2000 回（episode=1000）迄のデータをプロットしたものである。図 4.4 から、トレーニング回数 2000 の場合、改善前後の DDPG アルゴリズムの効果を全体として反映できていることが分かる。全体的な分析から見ると、両アルゴリズムに急激に変化の部分があり、かつ改良前の DDPG アルゴリズムは迅速に安定な状態に達することができたが、リターン値は比較的低いレベルにあった。改良された DDPG アルゴリズムは、各ラウンドにおいて高いリターン値レベルを達成することができ、改良された DDPG アルゴリズムがより短い時間で目標エリアまで走行することができることを示している。トレーニング回数がさらに増えると、反復ごとのスコアはさらに上昇せず、60000 前後のレベルで安定し、モデルがフィットした状態になったことを示している。

図 4.5 は、検証に用いたレースゲームの実装画面である。レーシングカーは経路を走行することができ、右下の十字がレーシングカーの方向を示している。レーシングカーが前に行くと「十字」の上の直線が青に表示され、ブレーキすると「十字」の下が青に表示され、左に行くと「十字」の左が青に表示され、右に行くと「十字」の右が青に表示される。トレーニング初期状態では、レーシングカーがコースから得ている情報は少なく、前方に壁が存在してもどの様に進めば

図 4.5　初期状態

良いか判断出来ず、壁にぶつかってもコースの中心方向へ戻ってはこないという挙動を取る。初期状態ではコースを走行することで得られるスコアも少ない傾向にある。

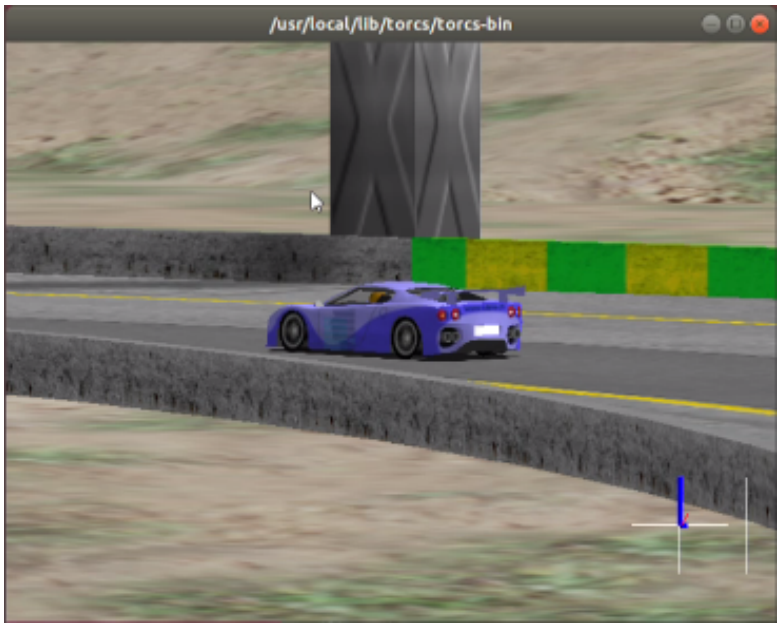

図 4.6　訓練後状態

図 4.6 は、本手法での訓練後の様子を示す。本手法では 50 回トレーニングを行うことで、レーシングカーがサーキットを 1 周することが可能となった。本手法は既存手法に比べてサーキットの道路状況情報により迅速に適応することができたと言える。レーシングカーのシミュレーショ

ン効果によって、このアルゴリズムが良好な効果を持つことを証明した。

4.3 本章のまとめ

本章では、シミュレーション実験を総合的に詳細に記述することにより、シミュレーションの環境を紹介した。その後、動的報酬関数に基づく DDPG ルート設計アルゴリズムと異なる重み経験に基づく再生 DDPG ルート設計アルゴリズムを紹介して実験分析を行い、そして実験データに対して分析を行い、本文で提案したアルゴリズムが比較的に良い効果を持つことを証明した。

第 5 章

まとめ

人工知能技術がレーシングカーの道路選択を補助する効果的な支援ソリューションとなっている。レーシングゲームでは、ゲーム制作の効率を上げるため，さらにレーシングカーのトレーニングの質とスピードを向上させるために、DDPG を最適化する。本文は DDPG アルゴリズムに基づいて以下の 2 種類の改善措置を行った。

1）動的報酬関数に基づく DDPG ルート設計アルゴリズム：DDPG ネットワークを訓練する異なる時期に、異なるタスクに対して異なる報酬関数設定した。トレーニング初期ではレースカーはコースに沿って走行できないため、距離中心線と衝突回数を大きなパラメータに設定した。トレーニングの後半では、レーシングカーはドリフトや衝突などの難易度の高い操作を行う必要があるため、この 2 つの係数を小さい係数に設定する必要がある。

2）異なる重みの経験再生に基づく DDPG ルート設計アルゴリズム：経験再生は学習に必要な経験の数を効果的に減らすことができる。経験を抽出するときに価値のある経験を限定的に抽出することで、モデルの適合性を向上させることができる。伝統的な経験の重みは同じであるため、本手法では経験に異なる重みパラメータを与え、これにより、経験に的を絞って訓練を行うことができる。

本手法は 改良 DDPG アルゴリズムを用い、実験データに対して分析を行い、本文で提案したアルゴリズムがトレーニングの速度の方面比較的に良い効果を持つことを証明した。本手法が実際のコースデザインの制作支援に役立つかの検証が今後の課題となる。

謝辞

本研究を進めるにあたり、多くのアドバイスや指導をしてくださり、相談に乗ってくれた渡辺先生、阿部先生には大変お世話になりました。第一稿の論文を添削して頂いた我彦拓磨先輩と助けていただきました陳方健先輩本当に心より感謝いたします。まだは辛い時期を支えてくれた孫さんと友人達や兄や、両親に深く感謝いたします。本当にありがとうございました。

参考文献

[1] 中川翔太永田裕一. DQN を用いたシューティングゲーム AI の制作. 情報部門学術講演会 2010, 2010.

[2] Inc Pixologic. Home of ZBrush. https://www.wikiwand.com/ja/DQN_(%E3%82%B3%E3%83%B3%E3%83%94%E3%83%A5%E3%83%BC%E3%82%BF). 参照: 2021.6.22.

[3] Casas N. Deep deterministic policy gradient for urban traffic light control[j]. *International Journal of Medical Informatics*, 2017.

[4] Inc Pixologic. Home of ZBrush. https://zhuanlan.zhihu.com/p/149771220?from_voters_page=true. 参照: 2021.6.22.

[5] S. Arumugam and C. Jermaine. Closest-point-of-approach join for moving object histories. In *22nd International Conference on Data Engineering (ICDE'06)*, pp. 86–86, 2006.

[6] Evangelos E Milios and S Hamid Nawab. Acoustic tracking from closest point of approach time, amplitude, and frequency at spatially distributed sensors. *The Journal of the Acoustical Society of America*, Vol. 87, No. 3, pp. 1026–1034, 1990.

[7] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107, 1968.

[8] R. Smierzchalski and Z. Michalewicz. Modeling of ship trajectory in collision situations by an evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 227–241, 2000.

[9] Sang-Min Lee, Kyung-Yub Kwon, and Joh Joongseon. A fuzzy logic for autonomous nav-

igation of marine vehicles satisfying colreg guidelines. *International Journal of Control, Automation, and Systems*, Vol. 2, No. 2, pp. 171–181, 2004.

[10] Yongqiang Zhuo and Grant E. Hearn. A ship based intelligent anti-collision decision-making support system utilizing trial manoeuvres. In *2008 Chinese Control and Decision Conference*, pp. 3982–3987, 2008.

[11] Jin-Hyeong Ahn, Key-Pyo Rhee, and Young-Jun You. A study on the collision avoidance of a ship using neural networks and fuzzy logic. *Applied Ocean Research*, Vol. 37, pp. 162–173, 2012.

[12] Rafał Szłapczyński and Joanna Szłapczyńska. Evolutionary sets of safe ship trajectories: Problem dedicated operators. In *International Conference on Computational Collective Intelligence*, pp. 221–230. Springer, 2011.

[13] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, Vol. 8, No. 3-4, pp. 279–292, 1992.

[14] M Stutzle. *Ant colony optimization*. Bradford Company, 2004.

[15] Lei Wang, Xu-Hui Xia, Jian-Hua Cao, Xiang Liu, and Jun-Wei Liu. Improved ant colony-genetic algorithm for information transmission path optimization in remanufacturing service system. *Chinese Journal of Mechanical Engineering*, Vol. 31, No. 1, pp. 1–12, 2018.

[16] S Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (March 1998), 1998.

[17] Paolo Magni, Silvana Quaglini, Monia Marchetti, and Giovanni Barosi. Deciding when to intervene: a markov decision process approach. *International Journal of Medical Informatics*, Vol. 60, No. 3, pp. 237–253, 2000.

[18] W. M. Liu, R. Li, J. Z. Sun, J. Wang, and P. M. Williams. Pqn and dqn: Algorithms

for expression microarrays. *Journal of Theoretical Biology*, Vol. 243, No. 2, pp. 273–278, 2006.

[19] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, Vol. 71, No. 7, pp. 1180–1190, 2008. Progress in Modeling, Theory, and Application of Computational Intelligenc.

[20] N. Casas. Deep deterministic policy gradient for urban traffic light control. 2017.

[21] N. Ketkar. Convolutional neural networks. *Springer International Publishing*, 2017.

[22] Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. Deep learning for reward design to improve monte carlo tree search in atari games. 2016.